Topology Management in Directional Antenna-Equipped Ad Hoc Networks

Ece Gelal, *Student Member*, *IEEE*, Gentian Jakllari, *Member*, *IEEE*, Srikanth V. Krishnamurthy, *Member*, *IEEE*, and Neal E. Young, *Member*, *IEEE*

Abstract—With *fully directional* communications, nodes must *track* and periodically update the positions of their discovered neighbors, so that communication with these neighbors is feasible when needed. If tracking fails, neighbors that move out of the directional footprint will need to be rediscovered. The tracking process introduces an overhead, which increases with the number of discovered neighbors. The overhead can be reduced if each node maintains only a subset of its neighbors; however, this may increase the hop count of paths between nodes, i.e., causes a *hop stretch.* In this work, we study the tradeoffs between node degree and hop stretch. We first design a topology control algorithm to optimize this tradeoff. For the purposes of this design, we assume that nodes perform circular directional transmissions to communicate with the nodes in their directional range; in this way, the network can be modeled as a unit disk graph (UDG). Given a UDG G, our algorithm finds a sparse subgraph G' with a maximum node degree of 6, and connecting each node pair u, v by a path of length $\mathrm{hops}_{G'}(u,v) = O(\mathrm{hops}_G(u,v) + \log\Delta)$, where Δ is the maximum degree in G, and $\mathrm{hops}_G(u,v)$ denotes the length of the shortest path between u and v in graph G. We show that this result is *near optimal*. Based on the insights gained from the above design, we next construct a more practical scheme, which integrates topology control with fully directional neighbor discovery and maintenance. The simulated performance of our practical scheme is only slightly worse than the theoretical optimal performance in terms of node degree and path stretch.

Index Terms—Topology control, directional antennas, fully directional communications, graph theory, ad hoc networks.

1 Introduction

In order to fully exploit possible benefits of spatial diversity and increased communication range with directional antennas, it is essential to shift to the *exclusive* usage of fully directional communications [16], [33], [48]. In addition to enabling better spatial reuse, fully directional communications also alleviate the phenomena of asymmetry in gain that arises when directional communications are used in conjunction with omnidirectional communications [18]. However, *neighbor discovery* and *neighbor maintenance* become challenging with fully directional communications due to the reduction in the angular range with the directional footprint.

For communicating with the discovered neighbors in a fully directional fashion, nodes need to acquire up-to-date information with regards to directions of their neighbors. This can be achieved with a periodic exchange of beacons; however, the overhead incurred due to this exchange may limit the performance improvements possible. Reducing the frequency of such control messages to bound the overhead is not an option; the frequency is dependent on how fast the nodes move or how fast the environment changes, and increasing the period between updates would cause nodes to have inaccurate neighborhood information. Our approach to limit this overhead is to use topology control for reducing node degrees in the network. We propose that nodes maintain logical connectivity only with a small subset of the discovered neighbors (say S); the neighbors that are not in S (and all other nodes in the network) are reached via multihop routes.

Reducing the node degrees can, however, have an adverse effect on the network connectivity and the hop count of paths between node pairs. Short paths are desirable, due to 1) maintaining low levels of packet loss (communication links in a multihop wireless network are error-prone and the probability of packet loss increases with route length), 2) avoiding high end-to-end delays (nodes compete for channel access at each hop), and 3) coping better with mobility (longer routes are more likely to fail when nodes move). Thus, while performing topology control to bound the node degree, we try to limit the end-to-end path stretch in the network.

Our first contribution in this work is the design of a topology control algorithm that finds the optimal tradeoff between **low node degree** and **low hop stretch**. Toward the design of this algorithm, we assume that nodes discover their neighbors by *circular directional communications* that are implicitly considered reliable [18]; we thus model the ad hoc network as a unit disk graph (UDG). We design the **Low Degree Spanner (LDS)** algorithm, which operates in a centralized fashion. Given the UDG $G = (V, E)$ representing all feasible links, LDS finds its sparse subgraph $G' = (V, E_{G'})$, wherein the maximum node degree is 6, and for each edge (u, v) in E, the vertices u, v are connected by a path of length $\mathrm{hops}_{G'}(u,v) = O(\mathrm{hops}_G(u,v) + \log\Delta)$

- *E. Gelal, S.V. Krishnamurthy, and N.E. Young are with the Department of Computer Science and Engineering, University of California, Riverside, Engineering Building Unit 2, Riverside, CA 92521. E-mail: {ecegelal, krish}@cs.ucr.edu.*
- *G. Jakllari is with BBN Technologies, 10 Moulton St., Cambridge, MA 02138. E-mail: gentian@bbn.com.*

Manuscript received 2 Nov. 2007; revised 24 June 2008; accepted 21 Oct. 2008; published online 5 Nov. 2008.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2007-11-0325. Digital Object Identifier no. 10.1109/TMC.2008.158.

(where Δ is the maximum degree of G, and $\text{hops}_{G'}(u,v)$ denotes the shortest path length from u to v in G'). We call the latter property as *bounded hop stretch*, and the graph satisfying this property as a *hop spanner*.

Our hop-spanner construction is facilitated by identifying disjoint groups in the ad hoc network, and interconnecting these groups via a backbone; this backbone is a degree-6 euclidean spanner of a designated set of nodes. These properties of the backbone are achieved by constraining the euclidean length and the angular positions of the included edges that are incident on these nodes. Finally, the intragroup connectivity is provided by forming balanced binary trees in every group (i.e., clique). The trees are then connected to the backbone while preserving the worst-case degree bound of 6 on all nodes. The properties of the euclidean spanner and tree structures aid our final results.

As our second contribution, we design the **Distributed LDS (D-LDS)** algorithm, which has the properties of LDS but provides a localized construction. With D-LDS, nodes make independent decisions on the links they maintain, based only on the knowledge of their two-hop neighborhoods. D-LDS operates on arbitrary networks, does not require synchronization among nodes, scales, and converges quickly to a connected topology with the desired properties.

To the best of our knowledge, the most closely comparable previous studies only achieve bounds on *one* of the metrics we consider. The approaches consist of 1) algorithms that find subgraphs with a constant hop stretch but do not impose a degree bound [12] and 2) algorithms that guarantee a low degree bound but do not examine hop stretch [22], [27], [47].

Third, drawing on the behavioral observations from D-LDS, we design a simpler but more practical scheme, **Directional Communications with Angular Topology Control (Di-ATC)**. Di-ATC relaxes the assumption that nodes have accurate two-hop neighborhood information; topology control decisions use the one-hop neighborhood information that is available via fully directional neighbor discovery and maintenance mechanisms. Di-ATC bounds node degree while avoiding high hop stretch, by means of allowing nodes to communicate with a subset of their neighbors, such that it is possible to angularly separate the directions in which the neighbors are maintained. To the best of our knowledge, Di-ATC is the first topology control algorithm for lightweight fully directional neighbor maintenance in a *mobile* ad hoc network.

We evaluate the performance of each of our distributed algorithms (D-LDS and Di-ATC) in terms of the node degree, hop stretch, and the incurred communication overhead. Our results indicate, as one might expect, that there is a tradeoff between the accuracy of nodes' neighborhood knowledge and the performance in terms of our metrics. With mode accurate information, nodes can form a sparser connected subgraph. Nevertheless, both our algorithms form topologies that satisfy the desired properties, given arbitrary input networks of different node density, area of deployment, and link reliability.

The rest of this paper is organized as follows: We present related previous work in Section 2. We describe the ad hoc network models used in the design of our algorithms in Section 3. We explain LDS and D-LDS algorithms in Section 4. In Section 5, we present Di-ATC and describe its integration with fully directional neighbor discovery and maintenance. We present the performance evaluation of our distributed protocols in various scenarios in Section 6. We conclude in Section 7.

TABLE 1
Comparison of Distributed Topology Control Algorithms in Terms of Key Features of D-LDS

Algorithm	Max. Node Degree	Hop-Stretch
[12]	Not bounded	Constant C, $C > 0$
[22]	6	Not bounded
[24]	8	Not bounded
[25]	$(k+1)^2 - 1$, k: number of cones	Not bounded
[27]	6	Not bounded
[40]	*(i)* 12 ; *(ii)* 9	Not bounded
[46]	$19+\lceil 2\pi/\alpha \rceil$, $\alpha \in (0, \pi/2)$	Not bounded
[47]	6	Not bounded
D-LDS	**6**	$\boldsymbol{O(1+\log(\Delta))}$

2 Related Work

We discuss related topology control efforts under three categories, focusing on the algorithms that bound the node degree, algorithms that bound the path stretch, and algorithms that utilize the benefits of antenna arrays. As mentioned before, there is a tradeoff between the sparseness of a topology and the lengths of the paths it contains.

Topology control algorithms that provide a low degree bound. Sparseness and *bounded degree* have been studied in several early efforts [3], [4], [11], [38]. More recently, Bose et al. proposed an algorithm to find a bounded-degree euclidean t-spanner of the *complete* input graph where $t \approx 10.02$; the degree bound is 27 [5]. These centralized algorithms are not practical for ad hoc deployments as they assume that all nodes can directly communicate with each other.

Several distributed algorithms defined proximity graphs of the given UDGs, and utilized the sparseness of these structures for guaranteeing degree bounds. However, although efficient in terms of a localized construction of a sparse topology, bounded-degree proximity structures are shown to yield $O(n)$ path stretch in terms of euclidean and topological distances (i.e., hop count) when the node distribution is not uniform [6], [11]. It is shown in [25] that no proximity graph with a constant degree bound contains the shortest (in terms of euclidean distance) path for all node pairs. This work also shows that for $O(1)$-spanner proximity structures (e.g., Yao Graph), the degree bound can be $O(n)$. Given this, the authors use a *Yao-Sink* structure to form a bounded-degree power spanner,[1] in which the degree bound can be made tighter by relaxing the spanning ratio (i.e., by changing the parameter k, see Table 1). Using the Yao-Sink graph, [26] proposes a localized algorithm to construct *directional* euclidean spanners having a maximum in-degree of 63. A more recent euclidean spanner algorithm achieves a tighter degree bound that is strictly greater than 8 [40]; Li et al. improved this bound to 8 in [24].

1. A power spanner yields paths, on which energy consumption is, at most, a constant times larger compared to minimum-energy paths in the UDG [25].

To date, the best theoretical bound on the node degree (that can be achieved with a distributed approach) is 6 and it is provided by the algorithms in [22], [27], and [47] (see Table 1). These approaches do not bound the hop stretch.

Topology control algorithms that provide a low hop stretch. Many of the algorithms cited above consider degree bound in conjunction with the path stretch. In these efforts, the goal is to limit the increase in euclidean length which is a consequence of reducing node degrees. However, a euclidean spanner is not necessarily a hop spanner. On a euclidean spanner, the route between a node pair u, v may have many short hops; in the worst case, the route visits all nodes along the straight line between them (no euclidean stretch but $O(n)$ hop stretch).

In [2], Alzoubi et al. proposed a localized algorithm for building a subgraph with a hop stretch of 3. The algorithm forms a backbone along which the node degree is bounded with very large values (specifically 295 for the *dominator*s and 7,384 for the *connector*s). Furthermore, these bounds do *not* hold in the final topology. Gao et al. proposed the first distributed algorithm that constructs a planar $O(1)$-spanner in terms of both euclidean and topological distance measures [12]. The constructed geometric spanner enables the use of local routing procedures such as geographical routing, even in existence of node mobility. However, the work does not consider the impact on the node degree. In [7], Burkhart et al. propose an algorithm, which, *given* a parameter t, constructs a euclidean t-spanner of the UDG; the authors indicate that these results are extendable to hop spanners with slight modifications. The work does not bound the node degree.

Summary. Topology control algorithms that we have discussed to this point were designed for use with omnidirectional communications. Most of these schemes analyzed worst-case bounds assuming static ad hoc networks, and they were not evaluated under conditions of mobility. Using these algorithms for topology control in MANETs that use directional (or fully directional) communications may not be efficient, since they do not account for the underlying physical model. Next, we describe the related work on directional communications.

Literature on directional antennas. The development of antenna arrays have inspired the design of novel MAC and routing protocols [10], [21], [17], [18], [31], [37], [43], [48]. A MAC protocol for use with directional antennas must help combat the problems of **asymmetry in range** and **deafness**. Korakis et al. showed that *asymmetry in range can be prevented by allowing only fully directional communications* (of control messages as well as data) [18]. Choudhury and Vaidya [9] and Li and Safwat [28] proposed methods to combat deafness (a node attempting to communicate with a node that is itself in the process of transmission). In [16], deafness is avoided via scheduled fully directional communications. The work describes why *resorting to omnidirectional transmissions or receptions degrades possible spatial reuse, reduces the possible extension in range, and causes asymmetry*.

Topology control using directional antennas. There is limited work to date for constructing topologies that improve the higher layer performance when directional antennas are used in an ad hoc network. Existing studies mostly focus on limiting the power consumption for extending network lifetime. In [13], Huang et al. propose an implementation of the "Cone-based topology control algorithm" (which was previously proposed for omnidirectional antennas in [20] and [47]), using sectorized antennas. The idea is to maintain fewer and closer neighbors in different antenna sectors. The algorithm assumes the use of a neighbor discovery procedure consisting of both directional and omnidirectional modes of communication; as mentioned before, this causes asymmetry and limited spatial reuse. In [14] and [15], Huang et al. constrain power consumption and reduce hop count by using multibeam directional antennas and adjusting the power intensity independently in each direction. In [30], power use is bounded with beam-switching antennas that may have nonuniform gains within the beamwidth. These efforts do not consider the cost of having high node degrees or the effects of mobility in the ad hoc network with fully directional communications.

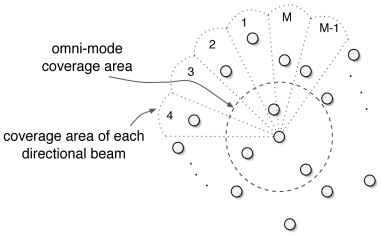


Fig. 1. Visualizing the circular sweeping of the directional footprint with an M-sector directional antenna.

There are recent studies on the use of directional antennas in wireless mesh networks (WMNs). Kumar et al. [19] propose topology control on WMNs by deploying multiple directional antennas at every node and properly orienting these antennas to create multiple low-interference topologies that are connected. Raman and Chebrolu present a mesh network implementation wherein each node is equipped with two high-gain directional antennas [32]; their measurements showcase the potential for using directional antennas to provide low-cost rural connectivity. These approaches do not consider the impact of mobility.

3 Model and Preliminaries

In the following, we describe how we model the ad hoc network that uses fully directional communications, and on which we execute the topology control algorithms in Sections 4 and 5.

3.1 Network Model for LDS/D-LDS

Both the centralized and distributed constructions to be described in Section 4 assume that nodes have up-to-date information with regards to their directional neighborhood. As we discuss in Section 5, there is an inherent challenge in discovering and maintaining the directional neighborhood in a node's fully directional range; nodes need to beamform in each other's direction and reliably exchange beacons [16], [33], [34]. For ease of the analysis in Section 4, we assume that the beacons are transmitted reliably (we consider the impact of collisions and antenna side lobes when we design Di-ATC in Section 5) in all directions. Given this, we model the directional neighborhood of a node to be circular (Fig. 1)

and thus, the topology created by the communications among nodes in the network to be a UDG.

Both LDS and D-LDS algorithms function on an *arbitrary* UDG[2] $G = (V, E)$, where a link exists between two nodes *iff* they are located within each other's communication range. Thus, while the maximum distance at which two nodes can successfully communicate is highly dependent on the environmental characteristics and the availability of line of sight, for simplicity of modeling, we assume that signal degradation occurs only due to path loss.

In our network model, each node has a distinct ID (e.g., MAC address). Our distributed scheme assumes that nodes can discover their *two-hop* neighborhoods;[3] efficient methods for this using omnidirectional communications are provided in [8] and [46]. We also assume that nodes are able to infer the relative positions of their neighbors [24], [47]. This can be achieved using the estimation techniques with antenna arrays for computing the angle of arrival (AOA) [43]. (Alternatively, nodes may be equipped with global positioning system (GPS) units to facilitate this capability. GPS has been used in various topology control studies including [22] and [23].)

3.2 Network Model for Di-ATC

The *near-optimal* (will be defined in Section 4) construction with the LDS and D-LDS algorithms are facilitated by modeling the communication network as a UDG. However, although popularly employed by most topology control studies to date, the UDG model does not accurately reflect the communication characteristics with directional antennas. In particular, we assume that the directional antennas have idealized footprints (i.e., no side lobes are present). More importantly, the operation of D-LDS relies on accurate neighborhood information and a connected topology. In practice, due to the characteristics of the fully directional neighbor discovery process, D-LDS may not perform efficiently in all scenarios.

In the network model for Di-ATC, we employ a beam-steering directional antenna which can target its boresight to an arbitrary direction. Our antenna system functions in two modes. When a node is searching for neighbors (to be discussed in detail in Section 5), the antenna functions as a switched beam antenna. It uses a fixed beamwidth Θ and can scan $2\pi/\Theta$ fixed directions. A node can either transmit or receive directionally at a given time. Once a node establishes a connection with a neighbor, it communicates with this neighbor by beamforming in its direction. Nodes continuously monitor the direction of arrival (DOA) or the AOA of their neighbors, from the signals they receive. A node updates its antenna weight coefficients to point its main lobe toward the target node. We assume that the time taken by an antenna to adapt its weight coefficients is negligible in comparison to the duration of a frame exchange.

2. In addition to being considered in the design of most topology control algorithms to-date [7], [24], [47], UDG model has also been used for modeling the neighborhood with directional beamforming [18] and with MIMO [41].

3. The two-hop neighborhood of node u is defined as the set of nodes N_1 that are neighbors of u, plus those that are neighbors of the nodes in N_1.

3.3 Preliminaries

We define the following two structures that are used by LDS.

Maximal independent set (MIS). Given a graph $G = (V, E)$, a subset M of V is an *independent set* if each edge in E is incident on at most one vertex in M. An MIS is an independent set such that for all vertices $v \in (V - M)$ the set $M \cup \{v\}$ is *not* independent; in other words, every vertex that is not in M is adjacent to some vertex in M. It is feasible to construct the set M in $O(n)$ time for UDGs, where $n = |V|$ [29], [42]. Distributed algorithms for finding an MIS are presented in [29] and [45].

Balanced binary tree. A nonempty binary tree is a *balanced* binary tree, if both its left and right subtrees (T_L and T_R) are balanced binary trees and $|height(T_L) - height(T_R)| \leq 1$. In a balanced binary tree consisting of N nodes, the length of the longest path from the root to a leaf node is $O(\log N)$.

Finally, we introduce the following notation that is used later.

Notation ($\angle(evf)$). When e and f are two edges incident on vertex v, we denote the angle between e and f at v as $\angle(evf)$.

4 Constructing a Hop Spanner with Bounded Degree

4.1 The Low-Degree Spanner (LDS)

LDS algorithm takes as input an arbitrary UDG $G = (V, E)$, as described in Section 3.

4.1.1 Efficiency of LDS

Before describing the LDS algorithm, we first show that the bounds achieved with its use are **near optimal**.

Lemma 1. *Given a connected UDG G, a connected subgraph G' with a maximum degree of less than five does not always exist.*

Proof. Consider the connected UDG with the star topology, where the central node u has five independent neighbors, each of which is exactly at a unit distance from u. If we remove the edge between u and *any* neighbor, i.e., make the degree of u less than five, the topology becomes disconnected. □

Lemma 2. *For a UDG G, any bounded-degree subgraph G' must have $\text{hops}_{G'}(u, v) = \Omega(\text{hops}_G(u, v) + \log \Delta)$ for some u, v, where $\text{hops}_G(u, v)$ represents the hop count of the shortest path between nodes u, v in G.*

Proof. Consider any degree-Δ node u and its neighbor set ν in G. In any subgraph $G' \subset G$ with maximum degree δ, let k be the maximum hop-distance from u to a node in ν. As there are at most δ^{k+1} nodes within k hops of u, $\Delta = |\nu| \leq \delta^{k+1}$, which implies $k \geq \log_\delta(\Delta) - 1$. □

4.1.2 Algorithm Description

LDS consists of three phases: 1) organizing the nodes in $G = (V, E)$ into distinct sets, 2) constructing a backbone that connects these sets, and 3) assembling the nodes (that are not on the backbone) in each set into balanced binary trees, which are then linked to the backbone.

Phase 1. *Creating groupings on G.* The purpose of this phase is to partition the nodes in V into disjoint groups such that any two nodes that belong to the same group have an

edge in E. To achieve this, we define a graph $G_{1/2} = (V, E_{1/2})$ such that there is an edge in $E_{1/2}$ between two nodes in V *iff* they are at most half of the unit distance apart from each other. Then, we find an MIS M of $G_{1/2}$ using the algorithm proposed in [42]. Nodes in the set M are called *dominators*; remaining nodes (nodes in $(V - M)$) are called *members*. Each member node is dominated by *one* dominator, which is within a distance of $1/2$ from it. The set of member nodes having a common dominator w form $group(w)$. Nodes in a group are within a unit distance from each other, i.e., they form a *clique*.

Phase 2. *Construction of the higher-tier backbone.* The purpose of this phase is to construct a bounded-degree backbone $H = (V_H, E_H)$ that interconnects separate groups. To achieve this, LDS: 1) defines a set V_H of "backbone nodes" and 2) selects the edges E_H that span V_H such that E_H is a *9-spanner* of V_H, with a maximum degree of 6. The decision on each edge is made based on its relative euclidean length and angle compared to other selected edges that are incident on the considered common node.

Step 1. V_H initially contains all dominators from Phase 1. Two groups are said to be *connected* if there is at least one edge between them in $G = (V, E)$. LDS chooses this edge (if there are multiple such edges, one of them is arbitrarily chosen) and marks the nodes at the end points of this edge as *backbone nodes*; these nodes are added to V_H. New backbone nodes are added to V_H until a pair of backbone nodes is designated for every pair of groups that is connected in G. Finally, in any group that has nonzero member nodes, if the dominator is the only backbone node, an arbitrary additional node s in this group is selected and added to V_H (this is necessary as will be seen later in Phase 3).

Step 2. LDS considers all edges (from $G = (V, E)$) between vertices in V_H, in the order of nondecreasing length. At each step, the considered edge $e = (u, v)$ is added to E_H *unless* there is an edge $f \in E_H$ such that f is incident on u or v ***and*** $\angle(evf) \leq 52$ degrees. (f was inserted before e implies that $d(f) \leq d(e)$, where $d(e)$ is the euclidean length[4] of e.) The phase terminates when all edges between the nodes in V_H have been thus considered.

This construction outputs a euclidean 9-spanner of the set V_H of backbone nodes. This is facilitated by the choice of backbone edges; the particular angular orientation and length of the selected edges prevents a euclidean stretch of greater than 9. We formally prove this property in Theorem 2.

Phase 3. *Connecting the remaining nodes to the higher-tier backbone.* In this phase, LDS completes the construction of the connected topology $G' = (V, E_{G'})$ with the desired properties. **1)** In every group, a balanced binary tree is formed from the nodes that are not in V_H. **2)** All trees are connected to the backbone such that the maximum degree on the backbone is preserved.

1) First, in every group that contains member nodes that are not on the backbone, LDS connects them with a rooted balanced binary tree $T(w)$. This construction is possible since every group is a clique. **2) Next**, LDS links the trees to the backbone as follows: Initially, $E_{G'}$ contains the edges in E_H. For each dominator w, if tree $T(w)$ exists (i.e., is not empty),

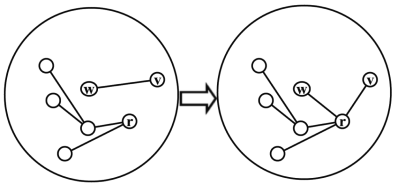


Fig. 2. Demonstration of how the balanced binary tree is connected to the backbone within the group.

LDS removes an arbitrarily selected edge (w, v) from $E_{G'}$ (Lemma 4 shows that this edge always exists), and adds two edges (r, w) and (r, v) instead (r is the root of $T(w)$). The edges in $T(w)$ are also added to $E_{G'}$. Fig. 2 depicts the procedure. In the figure, the edge (w, v) initially belongs to the backbone. The balanced binary tree is attached to the backbone at nodes w and v as discussed above. Note that this process preserves the degrees of w and v.

This phase leads to a hop spanner as we formally show in Theorem 3. The hop stretch is bounded in each group by the clique structure, it is also bounded on the backbone due to the properties of euclidean spanner and MIS.

Next, we show that the topology constructed by LDS has the desired properties in terms of node degree and hop stretch.

4.1.3 Analysis of LDS

Lemma 3. *Let $e = (u, v)$, $f = (u, w)$ be two edges incident on node u, and let $g = (v, w)$ be the edge across the angle $\angle(euf)$. If $d(f) < d(e)$ and $\angle(euf) \leq 52$ degrees, then $d(g) < d(e)$.*

Proof. Assume $d(g) > d(e)$. Then, $\angle(euf)$ is the largest angle in $\triangle(uvw)$, which implies $\angle(euf) \geq 60$ degree. However, $\angle(euf) \leq 52$ degrees, which contradicts the assumption; therefore, $d(g) < d(e)$. □

Lemma 4. *For a dominator u, if the number of nodes in $group(u)$ is greater than one, there is at least another backbone node v in $H = (V_H, E_H)$ such that $v \in group(u)$. Specifically, $(u, v) \in E_H$, where v is the backbone node closest to u.*

Proof. We prove the lemma by contradiction. Note that $group(u)$ contains at least two nodes in V_H by construction (Phase 2). Assume that $v \in V_H$ is the closest backbone node to dominator u,[5] and that edge $(u, v) \notin E_H$. This implies that there is an edge (u, w) or (v, z) in E_H (w, $z \in V_H$) that *blocks*[6] (u, v). However, the blocking edge cannot be (u, w) because this would imply $d(u, w) \leq d(u, v)$ which contradicts the assumption. Therefore, it must be that $(v, z) \in E_H$ blocks (u, v). Then, $d(v, z) \leq d(u, v)$ and $\angle(v, z)v(v, u) < 52$ degrees. By Lemma 3, $d(u, z) < d(u, v)$, which contradicts the assumption. □

Theorem 1. *The maximum node degree in G' is 6.*

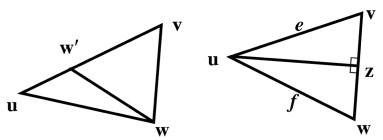
Proof. It suffices to show that $H(V_H, E_H)$ has a maximum degree of 6, because Phase 3 of the algorithm ensures that in each group

1. Member nodes that are not backbone nodes are assembled into balanced binary trees, such that the tree has a maximum node degree of 3,

4. Throughout this paper, we use the notation $d(e)$ for the euclidean length of edge e and $D(P)$ for the euclidean length of path P.

5. In case of ties, the backbone node with smaller ID is deemed closer.

6. We say that an edge e is "blocked" by an adjacent edge f *iff* $d(f) \leq d(e)$ and $\angle(evf) < 52$ degree.



(a) (b)

Fig. 3. Figure to aid the discussion of Theorem 2.

2. The degree of the root increases by 2 when connected to the backbone, but becomes at most 4, and
3. The degree of any node in H is preserved in constructing G' upon appending the balanced binary trees.

Then, this theorem holds due to the following lemma. □

Lemma 5. *The degree of any node in H is at most 6.*

Proof. Assume $\exists$ a node u in V_H with degree $deg(u) > 6$. Then, two of the edges that are incident on u *must* create an angle of at most $(360/7) < 52$ degrees. However, by construction, no two edges in H make an angle of less than 52 degrees. □

Theorem 2. *Each pair of backbone nodes u, v such that edge $e = (u, v) \in G$ are connected in H by a path P such that $D(P) \leq 9 \cdot d(e)$.*

Proof. The proof is by induction on the edges between the backbone nodes in G, in the order they are considered (whether to be a backbone edge) in constructing E_H. Consider one such edge $e = (u, v)$. If $(u, v) \in E_H$, then the theorem holds. If not, then $\exists$ an edge $f = (u, w) \in E_H$ such that $d(u, w) \leq d(u, v)$ and $\angle(euf) < 52$ degree. By Lemma 3 $d(v, w) < d(u, v)$ and thus, (v, w) was considered by LDS prior to (u, v). Consider path $P(v, u)$ formed with $P'(v, w)$ followed by edge (w, u). For induction, we postulate that, $\exists$ a path $P'(v, w)$ in E_H such that $d(P') \leq 9 \cdot d(v, w)$. We use this to show $d_H(u, v) \leq 9 \cdot d(u, v)$. Then, $d(P) \leq 9 \cdot d(v, w) + d(w, u)$. With this, it is enough to show that

$$d(u, w) + 9 \cdot d(v, w) \leq 9 \cdot d(u, v). \tag{1}$$

Or, equivalently,

$$\frac{d(u, w)}{d(u, v) - d(v, w)} \leq 9. \tag{2}$$

Let w' be a point on (u, v) s.t. $d(v, w') = d(v, w)$. Thus, the triangle $\triangle(vww')$ is isosceles with equal angles at w and w', as shown in Fig. 3a. Then, $d(u, w') = d(u, v) - d(v, w)$, and inequality (2) becomes

$$\frac{d(u, w)}{d(u, w')} \leq 9. \tag{3}$$

For any fixed $d(u, w)$ and a fixed angle $\angle(euf)$ at u, $d(u, w')$ is minimized when $d(u, v)$ is minimized. The smallest value $d(u, v)$ can take is $d(u, w)$, since inequality $d(u, w) \leq d(u, v)$ has to be satisfied. Then, the value of the left-hand side of inequality (3) is maximum when the denominator $d(u, w')$ is minimized, i.e., when $d(u, w) = d(u, v)$. This case is shown in Fig. 3b. The triangle $\triangle(uvw)$ is isosceles with equal angles at v and w. Let z be the midpoint of (v, w). By considering the right triangle $\triangle(uvz)$, we compute $d(v, z)$ and thereby $d(v, w)$ as

$$d(v, w) = 2 \cdot d(v, z) = 2 \cdot d(u, v) \cdot \sin\left(\frac{\angle(euf)}{2}\right). \tag{4}$$

Thus,

$$\begin{aligned} d(u, w) + 9 \cdot d(v, w) &= d(u, v) + 9 \cdot d(v, w) \\ &\leq d(u, v) + 18 \cdot d(u, v) \cdot \sin\left(\frac{\angle(euf)}{2}\right) \\ &\leq d(u, v) \cdot \left(1 + 18 \cdot \sin\left(\frac{\angle(euf)}{2}\right)\right) \\ &\leq 9 \cdot d(u, v). \end{aligned} \tag{5}$$

The last step in (5) is due to $\angle(euf) < 52$ degrees, which proves the theorem. □

Remark 1. The factor 9 can be improved to 8 by considering angles of $51.5 > 360/7$ degrees above instead of 52 degrees.

Lemma 6. *The number of groups connected to any given group g is $O(1)$.*

Proof. In Phase 1, dominators are chosen such that they are at least 1/2 units apart. Draw disks of radius 1/4 centered at each dominator whose groups are connected to g. These disks are disjoint and all of them lie within a circle of radius 2 centered at g's dominator. The number of such disks is $O(1)$. □

Corollary 1. *The number of backbone nodes in each group is $O(1)$. Therefore, the number of backbone nodes that are reachable from any backbone node is $O(1)$.*

Theorem 3. *For any two nodes u, v that are connected by a path P in $G = (V, E)$, there is a path P' in $G' = (V, E_{G'})$, such that $\text{hops}_{G'}(P') = O(\text{hops}_G(P) + \log \Delta)$. ($\Delta$ is the maximum node degree in G.)*

Proof. As each group forms a clique in G, each group has at most $\Delta + 1$ nodes. Consider the path P from u to v in G; let $g_1, g_2, g_3, \ldots, g_{k+1}$ be the sequence of groups that P traverses. P has at least k edges: for each $i = 1..k$, there is an edge from g_i to g_{i+1} in G. By construction, there is such an edge also in E_H for each $i = 1..k$. Let a_i and b_i be backbone nodes in g_i, such that a_i connects g_i to g_{i-1} and b_i connects g_i to g_{i+1}. This implies $d(b_{i-1}, a_i) \leq 1$, $d(a_i, b_i) \leq 1$, and $d(b_i, a_{i+1}) \leq 1$ in G. By Theorem 2, there is a path p_i from b_i to a_{i+1} and a path q_i from a_i to b_i in the backbone graph $H = (V_H, E_H)$, such that $D(p_i)$ and $D(q_i)$ are each ≤ 9. Construct P' from u to v in G' as follows: From u, traverse the edges in the balanced binary tree of u's group (g_1, w.l.o.g.) to get to the root. From the root, get to a_1. Then, traverse paths $p_1, q_2, p_2, q_3, \ldots, p_k, q_{k+1}$ to get from a_1 to b_{k+1}. From b_{k+1}, get to the root of the balanced binary tree in v's group (g_{k+1}); then traverse the tree to get from b_{k+1} to v. Traversal in each binary tree requires at most $\log_2 \Delta$ edges. As each of the paths p_i and q_i has euclidean length at most 9, they contain nodes from $O(1)$ groups (as they pass through $O(1)$ groups). By Corollary 1, each such path has $O(1)$ edges (connecting at

most all the backbone nodes in all visited groups). The number of these paths is $2k$. Additionally, the path may need to traverse $O(\log \Delta)$ edges in order to reach the backbone (via the binary balanced trees). Therefore, the hop count of path P' is $O(k + \log \Delta)$. □

Corollary 2. *For two nodes u, v that are connected by a path P in $G = (V, E)$, there is a path P' that connects them in $G' = (V, E_{G'})$ such that $D_{G'}(P') = O(D_G(P) + \log \Delta)$.*

Proof. The proof follows from Theorem 3 and from fact that each hop is of at most unit length. □

Lemma 7. *The time complexity of LDS is $O(n \log n)$.*

Proof. An MIS can be found in $O(n)$ time [29]. By Corollary 1, the number of edges on the backbone is $O(n)$ (the number of groups being $O(n)$ in the worst case). Sorting these edges takes $O(n \log n)$ time. Constructing each balanced binary tree takes $O(\Delta \log \Delta)$ time (requires sorting the nodes in a group with respect to their IDs). Therefore, the time complexity of LDS is $O(n \log n)$. □

4.2 Distributed Low-Degree Spanner (D-LDS)

In the following, we describe D-LDS, the distributed variant of LDS. With D-LDS, nodes make independent decisions on the links they maintain, based only on local (two-hop neighborhood) information. Using a constant number of broadcasts, D-LDS converges quickly to a connected topology with the near-optimal properties that we have been shown for LDS.

D-LDS has four phases.

Phase 1. *Finding the dominators and forming groups:* As with LDS, disjoint groups are formed in this phase. The *dominators* are identified using the distributed algorithm for finding an MIS from [45]. In brief, the algorithm works as follows: Initially, every node is colored white. A node selected as per some tie-breaking criterion (e.g., the node that has the lowest ID in its one-hop neighborhood) becomes black and declares itself a *dominator*, and it announces its transition with a one-hop broadcast. All of its white neighbors that receive this message become gray and declare themselves to be *dominated* by this dominator. The algorithm terminates when there are no more white nodes. By construction, each node has one dominator, since a node changes color only once. Upon termination of this construction, D-LDS defines the set consisting of a dominator and its associated gray nodes ("members") as a *group*. To ensure that all nodes in a group can communicate with each other, *in this phase*, nodes reduce their transmission power or increase their receiver sensitivity so that the achievable range is half of the actual maximum range.

Phase 2. *Identification of the backbone nodes:* In this phase, the backbone nodes are identified in a distributed fashion. Given the knowledge of nodes in its two-hop neighborhood, a dominator w designates nodes u, v to be backbone nodes corresponding to $group(w)$ and $group(z)$, **if** the following are satisfied:

1. w has a lower ID than z.
2. u, v are in different groups $(u \in group(w), v \in group(z), w \neq z)$.
3. u, v can communicate ($group(w)$ and $group(z)$ are connected).
4. A pair of backbone nodes has not already been assigned for these two groups.

Criterion 1 above avoids conflicts in the assignment of backbone node pairs by the corresponding two dominators. The dominator with a larger ID performs the assignment and notifies the other with a single unicast message; no synchronization is needed.

Before proceeding to Phase 3, every dominator w ensures that it is not the only backbone node in its group if $group(w)$ contains other nodes; it does so by designating an arbitrary node from $group(w)$ as a backbone node. As with LDS, this check ensures that if $group(w)$ contains nodes that will be connected to the backbone in Phase 4, then there is at least one link on the backbone such that both its end nodes are in $group(w)$.

Phase 3. *Construction of the backbone:* In this phase, each backbone node u participates in the distributed construction of a 9-spanner backbone. For this construction, u considers each backbone node v that is its one-hop neighbor, and examines whether the communication link (u, v) is "feasible." The decision regarding the feasibility of a link is made based on the estimated distances to the one-hop backbone neighbors. We define a distance estimation function that facilitates this construction:

Definition (distance function). *The distance function $\delta : P \rightarrow D$ maps power values to distance values; δ operates as follows:*

$$\begin{aligned} P_r(u,v) > P_r(u,z) &\Longrightarrow d(u,v) < d(u,z) \\ P_r(u,v) = P_r(u,z)\ \&\&\ ID(v) > ID(z) &\Longrightarrow d(u,v) < d(u,z) \\ P_r(u,v) < P_r(u,z) &\Longrightarrow d(u,v) > d(u,z). \end{aligned}$$

For its two neighbors v and z, node u estimates v to be closer than z if packets[7] received from v reflect greater power levels than those from z. Ties are broken based on IDs; the node with the higher ID is deemed closer. We assume that δ strictly increases with increasing received power; this is typically true for channels wherein the major source of degradation is path loss.

Definition (feasibility of a link). *Let Y denote the set of nodes that are closer to node u than v; nodes in Y reside inside the circle centered at u with a radius of $\hat{d}(u, v)$. Similarly, let Z denote the set of nodes that are closer to v than u. Initially, all links are unprocessed and unmarked:*

1. *A link (u, v) is* feasible, *if* ***both*** *of the following local conditions are satisfied:*
   a. *Every link (u, y_i) $(y_i \in Y)$ that would make an angle of < 52 degree with (u, v) must be marked as "unfeasible."*
   b. *Every link (v, z_i) $(z_i \in Z)$ that would make an angle of < 52 degree with (v, u) must be marked as "unfeasible."*

7. To prevent distance information from becoming stale, periodic updates will be necessary. We assume that the HELLO messages facilitate this process.

D-LDS (2-hop information of $G=(V,E)$ at each node)

```
{
 /* Identification of dominators & the groups */
1. Find a maximal independent set M of G=(V,E) in a distributed
   fashion (using algorithm in [26]) with 1/2 transmission radius.
2. // The elements of M are the dominators.
3. // [26] ensures that ∀ n∈(V-M) is dominated by a unique m∈M
4. // group(m): {m} ∪ {n : n dominated by m}, ∀m∈M

 /* Identification of backbone nodes -- executed at each dominator
    node w. */
5. S(w) ← w ; // S(w): the set of backbone nodes in group(w).
6. // Q(w)={ z∈M s.t. (w_k,z_k)∈ E && w_k∈group(w) && z_k∈group(z)} //
   Q(w): dominators whose groups are reachable from group(w)
7. Q(w) ← ∅ ; //this set is initially empty
8. for ∀ node u in group(w) do
9.    for (∀ link(u,v)∈G s.t. dominator(v)≠w && dominator(v)∉Q) do
10.      if ( ID(w) ≥ ID(dominator(v)))
11.         S(w) ← S(w) ∪ {u} ;
12.         Q(w) ← Q(w) ∪ {dominator(v)} ;
13.         send a msg to dominator(v) so that it executes lines 14,15 ;
14.         S(dominator(v)) ← S(dominator(v)) ∪ {v} ;
15.         Q(dominator(v)) ← Q(dominator(v)) ∪ {w} ;
16. if ( S(w) == {w} && (group(w)-S(w)≠∅) )
17.    s: any node from ( group(w)-S(w) ) ;
18.    S(w) ← S(w) ∪ {s};    // To ensure link (w,j) exists in line 53

/* Construction of backbone -- executed at each backbone node u. */
19. N_u ← nodes in 1-hop neighborhood of u ;
20. N(u,2) ← nodes in 2-hop neighborhood of u ;
21. for ∀ node v ∈ N_u do
22.    initialize edge status: status(u,v) ← UNPROCESSED ;
23. A ← N_u ;
24. a ← the closest neighbor of u (i.e. j=(u,a) is the shortest link
        among all (u,i) s.t. i∈A) ;
25. if ( j = (u,a) is the shortest link in N(u,2) )
26.    status(j) ← IN ;
27.    form link j=(u,a) ;
28.    send one broadcast message regarding status(=IN) of link j
29.    A ← A - {a} ;
30. while ( A is not empty ) do
31.    e=(u,v) ← next edge in A ; // status(e)= UNPROCESSED
32.    Y ← nodes {y}'s in N_u s.t. |u, y| ≤ |u,v| ;
33.    Z ← nodes {z}'s in N_v s.t. |v, z| ≤ |u,v| ;
34.    // u has knowledge of N_v.
35.    if ( ( for ∀ neighbor y ∈ Y s.t. ∠(gue)<52° where g=(u,y)
36.           status(g) == OUT OR Y == ∅ ) &&
37.         ( for ∀ neighbor z ∈ Z s.t. ∠(hve)<52° where h=(v,z)
38.           status(h) == OUT OR Z == ∅ ) ) then
39.       status(e) ← IN ;
40.       form link e=(u,v) ;
41.       send a broadcast msg regarding status(=IN) of link e
42.       A ← A - {e} ;
43.    elseif (∃ y ∈ Y s.t. ∠(gue)<52° where g=(u,y) && status(g)==IN
44.       OR ∃ z ∈ Z s.t. ∠(hve)<52° where h=(v,z) && status(h)==IN )
45.       status(e) ← OUT ;
46.       A ← A - {e} ;
47.       send a broadcast msg regarding status(=OUT) of link e

 /* Connecting the overall topology G'(V_G', E_G') -- executed at each
    dominator node w. */
48. R ← [ group(w)-S(w) ] ;
49. if ( R is not empty )
50.    Trigger the nodes in R to run Construct_BB_Tree(R) ;
51.    r ← Construct_BB_Tree(R) ; // return root of the tree
52.    (w,j) ← any edge in the backbone adjacent to w
53.    remove link (w,j) ;
54.    form link (w,r) ;
55.    send a message to r so that it forms the link (r,j) ;
} /* end of D-LDS */

Construct_BB_Tree (R)      // executed at each node in R
{
1. A ← sort nodes in R w.r.t. IDs (increasing order);
2. k ← my index in A ; //node A[k] connects to nodes A[2k], A[2k+1]
3. if ( 2k ≤ size(A) )              connect to A[2k] ;
4. if ( 2k+1 ≤ size(A) )      connect to A[2k+1] ;
5. return A[1];      // root of the balanced binary tree
}
```

Fig. 4. Distributed algorithm D-LDS constructs a low degree hop spanner with $O(1)$ broadcasts at each node.

A feasible *communication link is added to the topology.*

2. *For link* (u, v) *to be* unfeasible, *it is necessary and sufficient that* ***either*** *of the following two conditions is satisfied:*

   a. *There is a link* (u, y_i) $(y_i \in Y)$ *added to the topology, making an angle of less than 52 degrees with* (u, v).

   b. *There is a link* (v, z_i) $(z_i \in Z)$ *added to the topology, making an angle of less than 52 degrees with* (u, v).

   If link (u, v) *is unfeasible, both endpoints keep this information; nodes* u *and* v *do not communicate in the final topology.*

Each backbone node performs a local broadcast (which is in turn propagated to its two-hop neighborhood) upon deciding the status of its link with a backbone neighbor. The links in the topology at the end of this phase constitute the backbone.

Phase 4. *Finalizing the construction of the connected topology:* In every group, nodes that are not backbone nodes (if any) will form a balanced binary tree via a distributed procedure as shown in Fig. 4. Let the set of such nodes in $group(w)$ be $R(w)$. Tree construction in the group is triggered by the dominator w and it is performed concurrently at each node in $R(w)$. Each node carries out the following: 1) It sorts the nodes in $R(w)$ as per their IDs, in increasing order. (The sorted array is unique and is *the same* at all the nodes in $R(w)$, given that two-hop neighborhood information is available at all nodes.) 2) If it is at index k in the sorted order, it connects to nodes at indices $2k$ and $2k+1$ (if $2k$ or $2k+1$ do not exceed the number of elements in $R(w)$). This construction does not require any message exchange, and the tree is unique for a given set $R(w)$.

Without loss of generality, let node r be the root of the tree constructed in $group(w)$. r connects to the backbone, as w removes link (w, j) (the shortest backbone link adjacent to w by Lemma 4) and initiates the construction of the links (w, r) and (r, j). This procedure requires two unicast messages: from w to j for the removal of the link (w, j), and from w to r, to trigger the formation of link (r, j). Phase 4 terminates after this construction is complete for all groups.

D-LDS algorithm is depicted in Fig. 4.

Theorem 4. *The topology constructed by D-LDS has a maximum degree of 6.*

Proof. The proof follows from that of Theorem 1 as D-LDS emulates LDS in all phases. □

Theorem 5. *Let $G' = (V, E_{G'})$ be the topology constructed by D-LDS. For each pair of nodes u, v that were connected by a path P in G, they are connected by path P' in G', such that $\text{hops}_{G'}(u, v) = O(\text{hops}_G(u, v) + \log \Delta)$.*

Proof. First, we show that the backbone construction in Phase 2 of LDS and in Phase 3 of D-LDS output identical decisions with regards to the edges in the final topology; this is despite D-LDS working with only local information at the nodes. At the end of these phases, the constructed backbones are the same, as long as the input sets of backbone nodes are the same.

Recall that LDS sorts all possible edges between nodes in V_H in nondecreasing order, and considers one edge at a time. In D-LDS, a link $e = (u, v)$ is constructed, *only if* ***all*** edges that can *potentially* block it are marked *unfeasible.* This implies that, though the order in which the edges are processed may differ in LDS and D-LDS, the *same* decision is eventually made by both algorithms, regarding whether or not an edge is feasible or unfeasible.

Thus, by Theorem 3, G' has bounded hop stretch. □

Corollary 3. *For each pair of nodes u, v that had a path connecting them in G, there is a path P' connecting them in G', such that $D_{G'}(P') = O(D_G(P) + \log \Delta)$, where $D_G(P)$ denotes the euclidean length of path P.*

Proof. The proof follows from Theorems 3 and 5 and Corollary 2. □

Theorem 6. *For any input graph G, D-LDS constructs the final topology G' using $O(n)$ messages (in the worst case).*

Proof. Two-hop neighborhood information can be acquired at all nodes with $O(n)$ messages [8]. Message complexity of finding an MIS is $O(n)$ [45]. The number of backbone links is at most $O(n)$ (Lemma 7); thus, the total number of broadcasts for backbone construction is $O(n)$ (each backbone node performs one local broadcast upon deciding the status of a link). Balanced binary tree construction does not require message exchanges (two-hop neighborhood is already discovered). No broadcasts are necessary for linking the trees to the backbone. □

5 Topology Control with Directional Antennas

D-LDS algorithm provides insights that can be utilized in designing a more practical topology control scheme using directional communications. In particular, D-LDS demonstrates that when each node in the network maintains logical connectivity with its neighbors that are angularly separated, the hop stretch in the constructed *sparse* topology can be constrained.

Based on the key insights gained from the construction of D-LDS, we design a simpler but more practical topology control scheme, Di-ATC, for facilitating fully directional communications with bounded overhead. With Di-ATC, in addition to bounding the node degree without causing large hop stretch, we conform to the following design goals:

- *First*, it is essential that topology control is tightly knit with fully directional neighbor discovery and maintenance mechanisms. Omnidirectional transmissions or receptions must not be invoked in any phase of the algorithm.
- *Second*, topology control must preserve network connectivity. In particular, with fully directional neighbor discovery (as will be described later in this section), nodes may not acquire accurate information with regards to their neighbors. Consequently, aggressive topology control decisions (for low degree) may lead to network partitions or isolation.
- *Third*, since it is expensive (in terms of power, bandwidth, and time) to exchange directional control messages with neighbors, topology control must operate in a decentralized fashion and using information that is local and limited to the extent possible. Topology control can consequently scale with network size and in the presence of node mobility.

5.1 Angular Topology Control with Directional Antennas (Di-ATC)

With Di-ATC, given a degree bound K, nodes compute the angular separation that they will maintain among their neighbors in the formed topology. Nodes then communicate with a subset (selected based on angular separation) of their discovered neighbors. They track (i.e., exchange update beacons with) these neighbors periodically. This way, nodes proactively maintain the information in terms of "in which direction to beamform" in order to communicate with a neighbor. The decision at a node u, with regards to which neighbors it will track, is made based on its current degree, the current degrees and the angular positions of the neighbors that it is tracking (periodic messages should include the degree of the sender), and the degree bound K.

Given K, u decides to maintain a link with a discovered node v if *any* of the following conditions hold:

1. its current degree $(deg(u))$ is less than K.
2. $deg(u) = K$, but u does not have any other neighbors within $\Theta = \lceil 360/(K+1) \rceil$ degrees of the link (u, v). (If this case occurs, u has at least two other neighbors that are angularly apart by less than Θ. u will remove one of these links in favor of adding the link (u, v).)
3. $deg(u) = K$ and u has a neighbor z such that the angle between links (u, z) and (u, v) is less than or equal to Θ, but $deg(v)$ is strictly smaller than $deg(z)$ (in this case, u will attempt the removal of link (u, z) in favor of adding the link (u, v)).

The first condition implies that a node will maintain a discovered node as long has it has fewer neighbors than the degree bound. This behavior distinguishes Di-ATC from D-LDS. The condition is imposed for increasing the probability of maintaining connectivity; the decision helps to constrain the hop stretch as well. The second and third conditions are motivated by our design requirement of angularly separating the directions in which the neighbors are maintained. These criteria allow nodes to reach spatially separated areas with low hop stretch. A node u determines whether it has a conflict (based on the three criteria above) while adding the discovered neighbor v to its neighbor set.

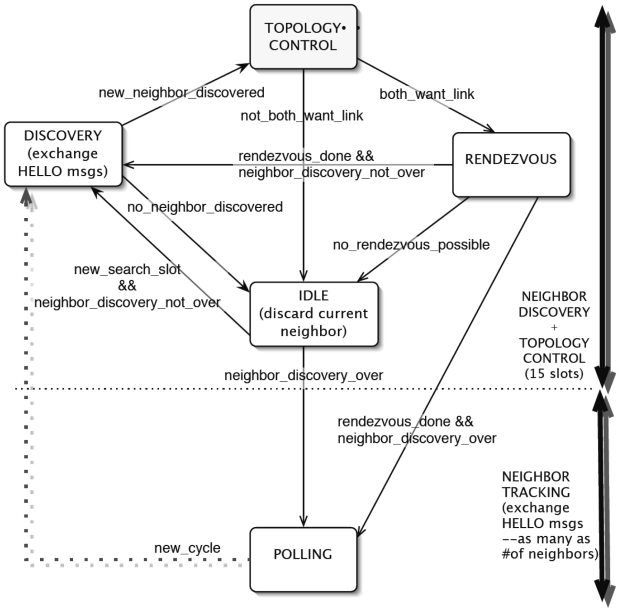


Fig. 5. State diagram depicting the stages of our integrated scheme.

Link (u, v) is formed only if *both* u and v agree on tracking each other independently, and reliably exchange topology control messages.

Note that these decisions do not rely on perfect directional neighborhood information or accurate knowledge of channel conditions. These heuristics are designed to support network connectivity; this is essential because nodes may easily lose their neighbors with mobility when omnidirectional communications are *not* invoked.

5.2 Integrating Di-ATC with Neighbor Discovery and Maintenance

We propose an integrated framework that allows seamless interactions among three functionalities; fully directional neighbor discovery, topology control on the discovered nodes, and directional maintenance of the neighbors that are chosen after topology control. As discussed earlier, the framework requires that nodes only *track* a subset of their directional neighbors. Di-ATC tries to find, given the available neighborhood information, the subset that offers good connectivity and short routes.

Our framework executes in cycles, as depicted in the state diagram in Fig. 5. All three of the aforementioned functionalities are invoked at each node in the network, in every cycle. Nodes are synchronized in terms of time slots that constitute these cycles. Synchronization is an essential characteristic for neighbor discovery and communications using fully directional communications [16], [34], [44]. Specifically, to facilitate a fully directional link, both end nodes must beamform toward each other at the same time. Synchronization can be achieved via solutions such as the method proposed in [36]; guard bands could also be employed to facilitate synchronization. We wish to point out that 1) only a coarse-grained synchronization at the frame level is needed, 2) only local synchronization among neighbors is needed, and 3) synchronization is a requirement of fully directional communications and not of our techniques in particular.

For fully directional neighbor discovery, each node beamforms toward a randomly chosen direction (or antenna sector, as was shown in Fig. 1) at specific time slots, and either transmits a HELLO packet in this direction or listens in the anticipation of receiving one. If a successful communication occurs, the node to first receive the HELLO beacon responds with one of its own to complete the handshake; the successful handshake implies that the node pair under discussion have discovered each other.

Directional neighbor discovery is a continuous process; nodes discover new neighbors as the topology changes due to mobility or due to variations in the environment. As shown in Fig. 5, several time slots in each cycle need to be allocated for the directional search for new neighbors. In particular, due to the probabilistic nature of the fully directional neighbor discovery process, it takes many cycles until nodes discover a fair percentage of their neighbors [16], [34], [44]. In our framework, we use Di-ATC with the neighbor discovery protocol proposed in [16].

We remark that other methods have been proposed for directional neighbor discovery; these approaches either include circular transmissions [18], or select a number of fixed beams [39], or apply transmit-only beamforming [34]. With all these schemes, the receptions during neighbor discovery are omnidirectional. As a consequence, these approaches may provide a smaller neighborhood than what is actually achievable with fully directional communications. These approaches also are prone to inefficiencies that arise due to asymmetry in range.

Di-ATC is invoked at both nodes (u and v) that discover each other. In the following time slot, the node pair exchanges the outcome of their Di-ATC execution. The new link (u, v) is established if both u and v *want* this link (based on the criteria described in the previous section) *and* can reliably exchange messages to agree on the link construction. Such local and dynamic decision-making renders the framework scalable under varying network conditions.

Tracking discovered neighbors is crucial with fully directional communications, especially when the nodes are mobile or when the environment changes dynamically. In our framework, nodes proceed to the neighbor tracking phase after neighbor discovery in every cycle (Fig. 5). Each pair of nodes that have a link will *rendezvous* at a common time and keep communicating on a periodic[8] basis. For this, we incorporate the polling phase proposed in [16]. If a node u does not repeatedly receive polling messages from w in the corresponding polling time, it assumes that the link is broken and removes w from its set of direct neighbors. This happens if w discovers a new neighbor z with a lower degree than u and abandons the link (w, u) to form a link with z.

Remark 2. The proposed scheme can also be implemented with beam-switching antennas; these antennas may be preferable as a less expensive alternative to fully adaptive arrays. In this case, as opposed to the *exact*

8. The frequency of these rendezvous instants depends on the frequency with which the network topology changes.

direction that maximizes the received signal power, nodes will determine the *closest* (in terms of angular separation from the intended direction) sector to the desired direction.

6 Performance Evaluation

We examine the average-case performances of our distributed algorithms D-LDS and Di-ATC[9] using simulations; we point out how our different design goals are reflected in their relative performances. We had analyzed the worst-case performance of D-LDS in Section 4; the performance evaluation in this section shows how its average-case performance compares with the worst-case bounds.

6.1 Simulation Setup

Simulation environment and communication model. We implemented D-LDS in a C++ simulator. Our simulator incorporates the conditions that were assumed for the design of D-LDS. In particular, 1) nodes have continuous access to information with regards to their two-hop neighborhood and 2) directional antenna footprints are ideal and no side lobes are formed.

We implemented Di-ATC in OPNET v.11 [1], in order to evaluate its performance under more realistic communication models. OPNET allows the design of antenna patterns having arbitrary gain and shape. In addition, one can modify the boresight of the created antenna with specific system calls and point the main lobe at an arbitrary point on the plane; the side lobes are created automatically, given the beamwidth of the main lobe and the maximum gain in the pointed direction. Upon each transmission in the pointed direction, the nodes within the directional footprint (that have their antennas pointed toward the transmitter) that receive the packet compute the signal-to-noise ratio (SNR) by considering the antenna gains and their positions relative to the transmitter. The corresponding bit error rate (BER) is computed based on this SNR value and the modulation scheme in use (we use BPSK in all scenarios). If the number of bit errors in the packet exceeds the error threshold that the decoder can accommodate, the packet is discarded at the MAC layer. This setup offers a fairly realistic communication model using directional antennas.

Input topology. We consider randomly generated topologies of varying densities in the area of deployment. When we simulate Di-ATC for a specific antenna beamwidth, the "unit" represents the communication range for that beamwidth. D-LDS is evaluated on static scenarios for various node densities and various network sizes. Di-ATC is evaluated at different densities, but due to the limitations in the simulation software, we do not simulate Di-ATC on very large networks. We also examine the performance of Di-ATC with node mobility (where nodes move according to the random waypoint mobility model).

Antenna-specific parameters for Di-ATC. All nodes use the same antenna model in a single simulation experiment; they all transmit with a fixed power in all experiments. The maximum gain in the main lobe is 20 dB (it is fixed in all simulations); the communication range varies with beamwidth. The directional range D is roughly a factor of $(360/\beta)^{1/\alpha}$ times larger than the omnidirectional range, where β is the antenna beamwidth and α is the path loss exponent; this model complies with that described in [35]. For the neighbor discovery phase, we use the method suggested in [16]. In particular, we reserve 15 time slots for neighbor discovery in every cycle. The cycles repeat continuously, and each simulation is run for 80 cycles.

With Di-ATC, nodes maintain logical connectivity with all discovered neighbors until the degree bound is reached (unlike the behavior with D-LDS where a node would not keep two neighbors that have a low angular separation). Therefore, in the topologies formed by Di-ATC, the average degree is close to the degree bound; this behavior is common in all scenarios. Because of this, in most of our experiments, we report results with a fixed beamwidth of 45 degree. We also impose a degree bound K of 6 unless specified otherwise, this value is motivated from the construction by D-LDS.

6.2 Parameters and Metrics

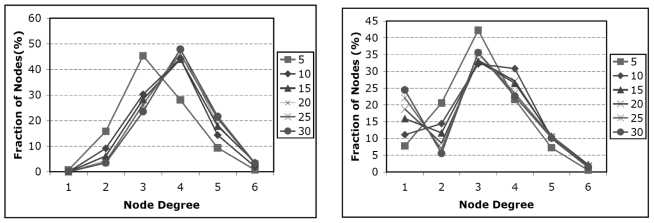
We test the dependence of our topology control algorithms on the following parameters:

1. *Node density.* We simulate both algorithms for various node densities in the region of interest. (Note that topology control is useful only in network deployments with moderate or high-densities.)
2. *Area of deployment.* We vary the size of deployment area.
3. *Antenna beamwidth.* We simulate Di-ATC using different antenna beamwidths (due to the assumption of circular transmissions that sweep the unit disk, the performance of D-LDS is independent of the antenna beamwidth).
4. *Speed of nodes.* We simulate Di-ATC in scenarios of low and high mobility.
5. *Degree bound.* Di-ATC imposes the degree bound a priori; we observe how our metrics specified below change when the bound is changed.

We quantify the performance in terms of the following metrics:

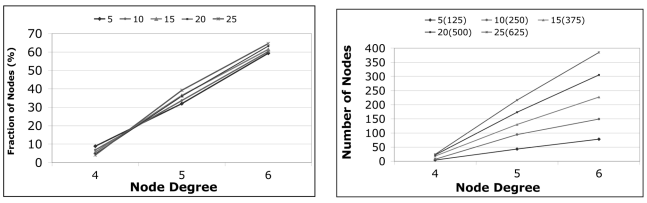
1. *Average node degree.* The average of all nodes' degrees in the network.
2. *Node degree distribution.* We measure the percentage of nodes in the network that have a certain degree.
3. *Average hop stretch.* For every node pair u, v that could communicate in the initial topology, we measure the hop count of the shortest path between them in the constructed topology. We define this value to be the hop stretch for the particular edge. An average is simply computed over all node pairs that can communicate in the scenario of interest. With Di-ATC, the hop stretch can be influenced by neighbor discovery; thus, we measure this metric relative to an *ideal* topology. "Ideal" topology implies that nodes can discover *all* nodes with which they can communicate fully directionally, *and* maintain *all*

9. We simulate Di-ATC, integrated with the fully directional neighbor discovery and the neighbor maintenance mechanisms.



(a) (b)

Fig. 6. Node degree distribution in the topology formed by D-LDS. (a) Distribution on the backbone. (b) Distribution in overall topology.



(a) (b)

Fig. 7. Node degree distribution in the topology formed by Di-ATC. (a) Fraction of nodes. (b) Number of nodes.

discovered neighbors. (This topology would include the shortest paths between all node pairs.)

4. *Maximum 95 percent hop stretch.* We quantify the 95 percent tail[10] of the hop stretch of all edges in the considered input graph.
5. *Hop stretch distribution.* The percentage of nodes having a certain hop stretch.
6. *Number of broadcasts.* We quantify the number of broadcasts performed during the execution of D-LDS. (As the MIS construction and acquiring the two-hop neighborhood are well-studied problems [29], [8], we do not include their broadcast overhead in quantifying this metric.)

We believe that these metrics capture the effectiveness of D-LDS and Di-ATC quantifying the *tradeoffs* between node degree and hop stretch.

6.3 Results and Discussion

In this section:

- We examine and compare the topologies formed by D-LDS and Di-ATC, in terms of *node degree* and *hop stretch*.
- We study the impact of changing the directional antenna beamwidth on the performance (in terms of node degree and hop stretch) of Di-ATC. (Since D-LDS assumes that nodes perform circular transmissions to sweep their neighborhood, its performance is independent of antenna beamwidth.)
- We study the node degree and hop stretch with Di-ATC in mobile scenarios.
- We quantify the message complexity of D-LDS with 1) reliable *and* 2) lossy links. With D-LDS, nodes' topology control decisions rely on accurate two-hop neighborhood knowledge; our objective here is to compute the cost of acquiring this information. Note that this is not required for Di-ATC operations.
- We study the impact of link losses (and the impact of resultant delay in making topology control decisions) on the connectivity of the topology formed by D-LDS.

Node degree performance. We know from Theorem 4 that the nodes in the topology constructed by D-LDS have a *maximum* degree of 6; we simulate D-LDS and observe the distribution of node degrees in the topology formed. Fig. 6 shows the degree distribution in the backbone and in the overall topology given networks of various node densities (from 5 to 30 nodes/unit^2 on the 6×6 region). As seen in Fig. 6a, almost half of the backbone nodes have degrees of at most 4 at node densities of 10 and above. (The UDG is quite sparse at a node density of 5; hence, nodes have lower degrees as compared to the other densities.) It is also important to note that not more than 4 percent of the nodes on the backbone have a degree equal to the worst-case bound of 6. As the balanced binary tree nodes have well-defined degrees (half of them have degree 3 and the other half, i.e., the leaves, have degrees of 1), the distribution of nodes in the final topology is biased toward lower degrees in Fig. 6b.

We also examine the degree distribution in the topologies generated by Di-ATC for input networks with different densities (nodes are deployed in a 5×5 unit area due to limitations in the ability of OPNET in performing large-scale simulations). We plot the percentage of nodes having a particular degree in Fig. 7a, and the actual number of nodes having that degree in Fig. 7b. With Di-ATC, most nodes have degrees equal to the maximum of 6 and overall, the node degrees are higher in comparison to the degrees with D-LDS. This is because Di-ATC allows a node u to keep a discovered neighbor z (irrespective of its angular separation with its existing neighbors), as long as the degree bound is not exceeded. On the other hand, the average degree with D-LDS is lower because 1) backbone nodes are not allowed to keep two neighbors that violate the angular separation constraint and 2) the nodes along the trees have a maximum degree of 4 (degree 4 occurring only at the root). With D-LDS, the average degree is slightly lower at higher densities, since the percentage of tree nodes increases. Overall, node density has little effect on the performance with both schemes. (Note here that, in these experiments with Di-ATC, we run the simulations long enough to facilitate discovery of the directional neighborhood; thus, the network converges and the effect of neighbor discovery on the topology control performance is reduced.)

Next, we measure the average node degree with Di-ATC using different antenna beamwidths and for varying degree bounds, with 500 nodes in the 10×10 unit area. In this experiment, a unit represents a radial range of 250 m. The results are shown in Figs. 8a and 8b. The average degree is slightly higher when wider-beam directional antennas are used. A lower node degree can be attributed to poor performance during neighbor discovery; it can also be a consequence of link removals with Di-ATC. The latter happens when nodes u, v discover each other and decide to

10. x percent tail of a set of values is the value that is bigger than x percent of the values in that set.

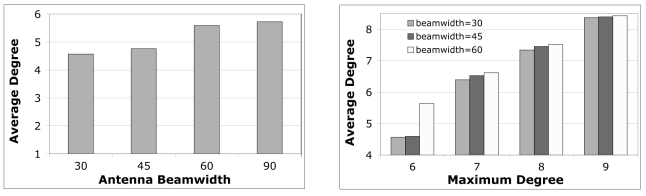


Fig. 8. Node degree performance of Di-ATC for varying parameters. (a) Average degree with different antenna beamwidths. (b) Average versus maximum degree.

form link (u, v), although they both have a degree equal to the upper bound. Consequently, both u and v will remove one of their existing links (say with nodes w and z, respectively); the degree of u and v remains the same, but the degrees of both w and z decrease by 1.

Hop stretch performance. In addition to ensuring low node degrees, both our schemes yield **low hop stretch**. The topology constructed by D-LDS, for all considered networks of high density (as we plot in Fig. 9a), is a hop spanner with a worst-case maximum 95 percentile hop stretch of 12. In the same figure, we also show that the average hop stretch is close to 6 for *all* network sizes that were considered. These results demonstrate that the performance of D-LDS *scales* in moderately sized to large networks. We also measure the average and the 95 percentile hop stretch as a function of node density (the area of deployment is fixed at 6×6 units). We observe that the hop stretch is even lower at the moderate node densities that are more likely in ad hoc network deployments (Fig. 9c). In Figs. 9b and 9d, we show the percentage of node pairs that have a specific hop stretch in a given network.

In Fig. 9c, we also present the average and maximum hop stretch with Di-ATC given networks with different node densities. In this figure, one can visualize how these results compare with the results from D-LDS. Both the maximum and the average hop stretch with Di-ATC are lower compared to the corresponding results with D-LDS. This is due to the higher average degree in the topology formed by Di-ATC (Figs. 6 and 7). Hop stretch increases in denser deployments, as the output topology includes a smaller ratio of the total number of links that are in the ideal topology. This increase is less visible with Di-ATC, as nodes maintain a larger subset of their discovered neighbors than they do when they execute D-LDS.

Next, we examine the impact of angular separation on the hop stretch with Di-ATC by varying the imposed degree bound. We experiment with different antenna beamwidths; the hop stretch with each beamwidth is computed relative to the corresponding *ideal* topology (discussed earlier). We present the maximum 95 percent and average values of hop stretch in Fig. 10. As expected, the hop stretch is less severe when the degree bound requirement is less strict (Fig. 10a).

Fig. 10b shows that for a given degree bound the hop stretch is higher when narrower directional beams are used. Probability of neighbor discovery increases with a larger beam. Consequently, nodes have a greater degree of freedom in choosing the angularly separated subset of their discovered neighbors. Furthermore, nodes' neighborhood in the ideal topology corresponding to a narrower beamwidth includes more links. This is because, due to the larger radial range with a narrower-beam directional antenna, a node can potentially reach (though it may not discover) more neighbors. As a consequence, the hop stretch in the output topology G' is higher, as more links that make up the shorter paths in the ideal topology do not exist in G'.

Fig. 10c shows that more than half of the paths in the topology formed by Di-ATC have a stretch of less than or

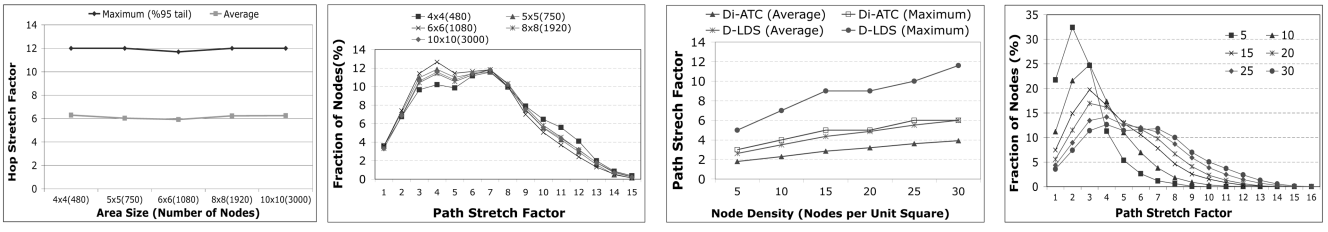


Fig. 9. Hop stretch for the topologies generated by D-LDS and Di-ATC. (a) Average and maximum hop stretch at high density (30 $\mathrm{nodes/unit}^2$). (b) Hop stretch distribution at high density (30 $\mathrm{nodes/unit}^2$). (c) Average and maximum hop stretch at various densities. (d) Hop stretch distribution at various densities.

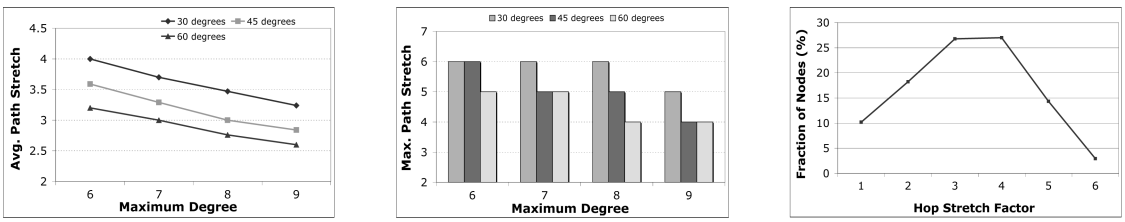
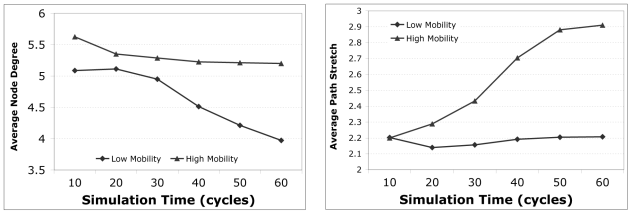
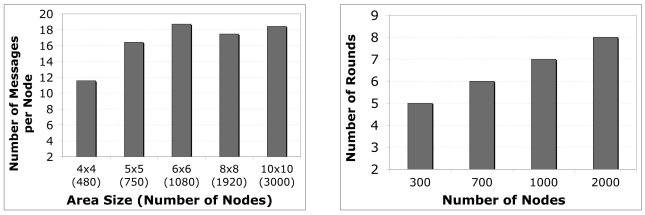


Fig. 10. Hop stretch performance of Di-ATC. (a) Average hop stretch. (b) Maximum hop stretch. (c) Hop stretch distribution (45 degree).



(a) (b)

Fig. 11. Average node degree and hop stretch in the topologies generated by Di-ATC in existence of slow and fast node mobility. (a) Average degree versus time with node mobility. (b) Average hop stretch versus time with node mobility.



(a) (b)

Fig. 12. Message and time complexity with D-LDS for various topologies. (a) Message complexity of D-LDS. (b) Time convergence of D-LDS.

equal to 4; this result can be ascribed to the relatively high node degrees resulting with Di-ATC unlike with D-LDS (Figs. 6 and 7).

Performance in mobile scenarios. We simulate Di-ATC in two mobile scenarios with the random waypoint mobility model. Nodes choose a random speed between (0, 10] m/s; this represents a scenario in which the topology changes are slow. In a second scenario, nodes choose speeds in (10, 20] m/s for a faster-changing topology. In both scenarios, nodes pause for 1 s at each intermediate destination point. The 1 s pause time corresponds to a duration of 2.5 cycles (recall our discussion of cycles in Section 5). As nodes' coordinates are dynamic due to mobility, we compute the hop stretch for the mobile case in a different fashion. At particular time instants, we find the shortest path between two nodes (u, v) in the snapshot of the constructed topology at that time instance, and compare it with the shortest path (between u, v) in the ideal topology consisting of the links that could potentially exist at that time. (As before, the ideal case assumes that the complete neighborhood is discovered and all links are maintained.)

Fig. 11 shows the average degree and hop stretch in a mobile network, at arbitrarily selected cycle instants in time during a simulation run. Fig. 11a shows that the average degree does not fluctuate much after stabilizing at cycle 40 when mobility is low. With higher node mobility, the average degree drops after the first half of the simulation, as nodes lose their neighbors due to high relative speeds (it becomes very hard to track neighbors). Note, however, that these high speeds are not common in typical network deployments. In fact, at these extremely high speeds, use of steerable directional antennas with narrow beams is challenging.

We observe that the hop stretch also stabilizes around cycle 40 (Fig. 11b). The initial period until stabilization is spent for discovering neighbors. The hop stretch exhibits a slight increase after cycle 30 in the case of fast mobility, due to the decrease in node degrees. The low hop stretch can be attributed to the angular separation among neighbors of a node, which helps Di-ATC construct sparse but well-spread topologies.

Communication and time complexity of D-LDS. We showed in Section 4 that the number of messages required by D-LDS is $O(n)$ i.e., $= cn$, where c is a constant. The simulations, however, can provide an estimate of the hidden constant c. To quantify c, we generate enough nodes such that there are approximately 30 nodes in every square unit; this is considered a dense ad hoc network. Fixing this density, the input network is varied from 480 nodes in a 4×4 unit area (a moderate-size ad hoc network) to 3,000 nodes in a 10×10 unit area (a large ad hoc network). We measure the total number of broadcasts used by D-LDS in constructing the final topology, and divide it by the number of nodes. The results are depicted in Fig. 12a; they show that the measured estimate of the constant stabilizes to between 18 and 19 and does not change even when the number of nodes is as high as 3,000.

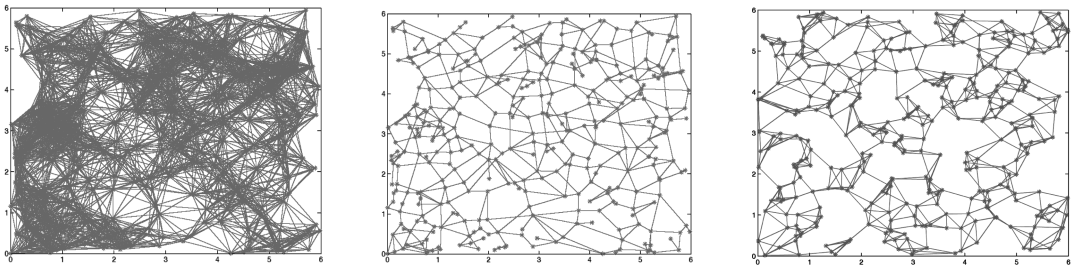
We also examine the time it takes until the backbone converges to a 9-spanner. This provides us with an idea of the parallelism possible with D-LDS. We define convergence time in terms of "the number of rounds," where a round corresponds to the duration of transmission of a single message. Since with D-LDS, edges can be inserted in parallel, we expect the time taken for convergence to be much smaller than the worst-case $O(n)$ bound. To corroborate this expectation, we construct topologies of varying densities in a 6×6 unit area. We depict the results in Fig. 12b. We observe that even with a large number of nodes (as high as 2,000), convergence takes eight rounds. This demonstrates the high degree of parallelism that is possible with D-LDS.

Connectivity with D-LDS with lossy links. The packet losses at the link layer can delay the establishment of edges with D-LDS. We investigate how the connectivity of the network is affected by this delay. In other words, we study the convergence of D-LDS in terms of forming the final graph. To understand this, we perform the following experiment. One thousand nodes are placed in a 6×6 unit area and trace the connectivity of the backbone at different stages of D-LDS execution; at each stage, note that only a percentage of the final set of edges is established. We investigate the impact on the backbone connectivity only; the binary balanced trees are constructed by nodes that are within the one hop reach of each other, and hence, do not affect the connectivity of the other nodes beyond their group.

The results of the experiment are shown in Table 2. Note that D-LDS provides connectivity even when just a

TABLE 2
Connectivity at Different Stages of the D-LDS Execution

Established Edges (%)	Connectivity (%)
50	62.9
60	82.2
70	96
80	99



(a) (b) (c)

Fig. 13. Visualization of the input topology as a UDG and the topologies generated by D-LDS and Di-ATC. (UDG consists of 360 nodes in a 6×6 area.) (a) Input topology. (b) Topology generated by D-LDS. (c) Topology generated by Di-ATC.

subset of edges is established; connectivity is guaranteed even if only 70 percent of the edges are established.

Summary. we have evaluated the performance of our distributed topology control algorithms D-LDS and Di-ATC, in various scenarios. We have showed their performances scale in terms of node density, network size, and area of deployment. Fig. 13 illustrates a randomly[11] generated network, and the sparse topologies constructed by D-LDS and Di-ATC on this network. The initial graph consists of 360 nodes in a 6×6 unit area. The unit with D-LDS is as defined in the beginning of this section; with Di-ATC we use 45 degree-beamwidth directional antennas and map its directional range to a unit distance.

7 Conclusions

Nodes in a fully directional ad hoc network need to continuously track their neighbors for maintenance. In this paper, we proposed topology control strategies in order to reduce the overhead incurred due to this process.

We first designed the LDS algorithm that finds a sparse subgraph of a given arbitrary UDG G, such that the maximum degree is **6** and the maximum hop stretch is $O(1 + \log \Delta)$, where Δ is the maximum degree in G. We show that this result is **near optimal**.

Next, we designed the distributed, localized algorithm D-LDS, which forms a sparse topology having the same attractive properties as LDS using a linear number of broadcasts per node.

Finally, we relaxed the idealized assumptions we made to facilitate the design of LDS and D-LDS, and we designed a more practical, considerably simpler topology control scheme Di-ATC. Di-ATC integrates fully directional neighbor discovery with the topology control decisions, and neighbor maintenance mechanism is applied only to a selected subset of discovered neighbors. We simulated the integrated framework in practical settings and studied the average performance to understand the tradeoff between node degree and hop stretch.

11. While our simulations function on input topologies where we generate nodes randomly using a uniform distribution, the bounds that were derived previously hold for **arbitrary** UDGs.

References

[1] *Opnet User's Documentation,* http://www.opnet.com, 2008.

[2] K. Alzoubi, X. Li, Y. Wang, P. Wan, and O. Frieder, "Geometric Spanners for Wireless Ad Hoc Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 14, 2003.

[3] S. Arya, G. Das, D.M. Mount, J.S. Salowe, and M. Smid, "Euclidean Spanners: Short, Thin, and Lanky," *Proc. 27th ACM Symp. Theory of Computing (STOC '95),* 1995.

[4] S. Arya and M. Smid, "Efficient Construction of a Bounded Degree Spanner with Low Weight," *Proc. Second Ann. European Symp. Algorithms (ESA '94),* 1994.

[5] P. Bose, J. Gudmundsson, and M. Smid, "Constructing Plane Spanners of Bounded Degree and Low Weight," *Proc. 10th Ann. European Symp. Algorithms (ESA '02),* 2002.

[6] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick, "On the Spanning Ratio of Gabriel Graphs and β-Skeletons," *SIAM J. Discrete Math.,* 2001.

[7] M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger, "Does Topology Control Reduce Interference," *Proc. ACM MobiHoc,* 2004.

[8] G. Calinescu, "Computing 2-Hop Neighborhoods in Ad Hoc Wireless Networks," *Proc. Int'l Conf. Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW '03),* 2003.

[9] R. Choudhury and N. Vaidya, "Deafness: A MAC Problem in Ad Hoc Networks When Using Directional Antennas," *Proc. 12th IEEE Int'l Conf. Network Protocols (ICNP '04),* 2004.

[10] R. Choudhury, X. Yang, N. Vaidya, and R. Ramanathan, "Using Directional Antennas for Medium Access Control in Ad Hoc Networks," *Proc. ACM MobiCom,* 2002.

[11] D. Eppstein, "Spanning Trees and Spanners," technical report, 1996.

[12] J. Gao, L.J. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Geometric Spanner for Routing in Mobile Networks," *Proc. ACM MobiHoc,* 2001.

[13] Z. Huang, C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Topology Control for Ad Hoc Networks with Directional Antennas," *Proc. 11th IEEE Int'l Conf. Computer Comm. and Networks (ICCCN '02),* 2002.

[14] Z. Huang and C.-C. Shen, "Topology Control with Directional Power Intensity for Ad Hoc Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '04),* 2004.

[15] Z. Huang and C. Shen, "Multibeam Antenna-Based Topology Control with Directional Power Intensity for Ad Hoc Networks," *IEEE Trans. Mobile Computing,* vol. 5, no. 5, May 2006.

[16] G. Jakllari, W. Luo, and S.V. Krishnamurthy, "An Integrated Neighbor Discovery and MAC Protocol for Ad Hoc Networks Using Directional Antennas," *IEEE Trans. Wireless Comm.,* vol. 6, no. 3, 2007.

[17] Y. Ko, V. Shankarkumar, and N. Vaidya, "Medium Access Control Protocols Using Directional Antennas in Ad Hoc Networks," *Proc. IEEE INFOCOM,* 2002.

[18] T. Korakis, G. Jakllari, and L. Tassiulas, "A MAC Protocol for Full Exploitation of Directional Antennas in Ad Hoc Wireless Networks," *Proc. ACM MobiHoc,* 2003.

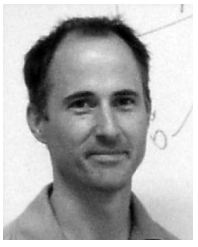
[19] U. Kumar, H. Gupta, and S.R. Das, "A Topology Control Approach to Using Directional Antennas in Wireless Mesh Networks," *Proc. IEEE Int'l Conf. Comm. (ICC '06),* 2006.

[20] E. Li, J. Halpern, P. Bahl, Y. Wang, and R. Wattenhofer, "Analysis of a Cone-Based Distributed Topology Control Algorithm for Wireless Multi-Hop Networks," *Proc. ACM Symp. Principles of Distributed Computing (PODC '01),* 2001.
[21] G. Li, L. Yang, W. Conner, and B. Sadeghi, "Opportunities and Challenges for Mesh Networks Using Directional Antennas," *Proc. First IEEE Workshop Wireless Mesh Networks (WiMesh '05),* 2005.
[22] N. Li, J.C. Hou, and L. Sha, "Design and Analysis of an MST-Based Topology Control Algorithm," *Proc. IEEE INFOCOM,* 2003.
[23] X. Li, G. Calinescu, and P. Wan, "Distributed Construction of a Planar Spanner and Routing for Ad Hoc Wireless Networks," *Proc. IEEE INFOCOM,* 2002.
[24] X. Li, W. Song, and W. Wang, "A Unified Energy-Efficient Topology for Unicast and Broadcast," *Proc. ACM MobiCom,* 2005.
[25] X. Li, P. Wan, and Y. Wang, "Power Efficient and Sparse Spanner for Wireless Ad Hoc Networks," *Proc. 10th IEEE Int'l Conf. Computer Comm. and Networks (ICCCN '01),* 2001.
[26] X. Li, P. Wan, Y. Wang, and O. Frieder, "Sparse Power Efficient Topology for Wireless Networks," *Proc. 35th Ann. Hawaii Int'l Conf. System Sciences (HICSS '02),* 2002.
[27] X. Li, Y. Wang, P. Wan, and O. Frieder, "Localized Low Weight Graph and Its Applications in Wireless Ad Hoc Networks," *Proc. IEEE INFOCOM,* 2004.
[28] Y. Li and A.M. Safwat, "Efficient Deafness Avoidance in Wireless Ad Hoc and Sensor Networks with Directional Antennas," *Proc. Second ACM Int'l Workshop Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN '05),* 2005.
[29] M. Min, F. Wang, D. Du, and P.M. Pardalos, "A Reliable Virtual Backbone Scheme in Mobile Ad-Hoc Networks," *Proc. First IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems (MASS '04),* 2004.
[30] V. Namboodiri, L. Gao, and R. Janaswamy, "Power Efficient Topology Control for Wireless Networks with Switched Beam Directional Antennas," *Proc. Second IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems (MASS '05),* 2005.
[31] A. Nasipuri, S. Ye, J. You, and R. Hiromoto, "A MAC Protocol for Mobile Ad Hoc Networks Using Directional Antennas," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '00),* 2000.
[32] B. Raman and K. Chebrolu, "Design and Evaluation of a New MAC Protocol for Long-Distance 802.11 Mesh Networks," *Proc. ACM MobiCom,* 2005.
[33] R. Ramanathan, "On the Performance of Ad Hoc Networks with Beamforming Antennas," *Proc. ACM MobiHoc,* 2001.
[34] R. Ramanathan, J. Redi, C. Santivanez, D. Wiggins, and S. Polit, "Ad Hoc Networking with Directional Antennas: A Complete System Solution," *IEEE J. Selected Areas in Comm.,* vol. 23, 2005.
[35] R. Ramanathan, M. Takai, and N. Vaidya, "Directional Antenna Systems for Ad Hoc Networks," *Proc. ACM MobiHoc,* tutorial, 2003.
[36] K. Romer, "Time Synchronization in Ad Hoc Networks," *Proc. ACM MobiHoc,* 2001.
[37] S. Roy, D. Saha, S. Bandyopadhyay, T. Ueda, and S. Tanaka, "A Network-Aware MAC and Routing Protocol for Effective Load Balancing in Ad Hoc Wireless Networks with Directional Antenna," *Proc. ACM MobiHoc,* 2003.
[38] J.S. Salowe, "Euclidean Spanner Graphs with Degree Four," *Discrete Applied Math.,* vol. 54, no. 1, 1994.
[39] R. Santosa, B.-S. Lee, C. Yeo, and T. Lim, "Distributed Neighbor Discovery in Ad Hoc Networks Using Directional Antennas," *Proc. Sixth IEEE Int'l Conf. Computer and Information Technology (CIT '06),* 2006.
[40] W. Song, Y. Wang, and X. Li, "Localized Algorithms for Energy Efficient Topology in Wireless Ad Hoc Networks," *Proc. ACM MobiHoc,* 2004.
[41] K. Sundaresan, R. Sivakumar, and M. Ingram, "A Fair Medium Access Control Protocol for Ad-Hoc Networks with MIMO Links," *Proc. IEEE INFOCOM,* 2004.
[42] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms.* The MIT Press, 2001.
[43] M. Takai, J. Martin, R. Bagrodia, and A. Ren, "Directional Virtual Carrier Sensing for Directional Antennas in Mobile Ad Hoc Networks," *Proc. ACM MobiHoc,* 2002.
[44] S. Vasudevan, J. Kurose, and D. Towsley, "On Neighbor Discovery in Wireless Networks with Directional Antennas," *Proc. IEEE INFOCOM,* 2005.
[45] P. Wan, K. Alzoubi, and O. Frieder, "Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks," *Proc. IEEE INFOCOM,* 2002.
[46] Y. Wang and X. Li, "Localized Construction of Bounded Degree and Planar Spanner for Wireless Ad Hoc Networks," *Proc. Joint Workshop Foundations of Mobile Computing (DIALM-POMC '03),* 2003.
[47] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, "Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks," *Proc. IEEE INFOCOM,* 2001.
[48] S. Yi, Y. Pei, and S. Kalyanaraman, "On the Capacity Improvement of Ad Hoc Wireless Networks Using Directional Antennas," *Proc. ACM MobiHoc,* 2003.

Ece Gelal received the BSc degree in telecommunications from Sabanci University, Turkey, in 2004 and the MSc degree in computer science and engineering from the University of California (UC), Riverside, in 2006. She is currently a PhD candidate in the Department of Computer Science and Engineering, UC Riverside. Her research interests include cross-layer design and evaluation of MAC and routing layer mechanisms for multihop wireless networks. She is a student member of the IEEE.

Gentian Jakllari received the bachelor's degree in computer science from the University of Ioannina, Greece, and the MSc and PhD degrees in computer science from the University of California, Riverside. He is a scientist in BBN Technologies, Cambridge, Massachusetts. His research interests include the field of wireless networking with an emphasis on the design, evaluation, and implementation of cross-layer protocols and algorithms. He is a member of the IEEE.

Srikanth V. Krishnamurthy received the PhD degree in electrical and computer engineering from the University of California, San Diego, in 1997. From 1998 to 2000, he was a research staff scientist in the Information Sciences Laboratory, HRL Laboratories, LLC, Malibu, California. He is currently an associate professor of computer science in the Department of Computer Science and Engineering, University of California, Riverside. His research interests include CDMA and TDMA technologies, medium access control protocols for satellite and wireless networks, routing and multicasting in wireless networks, power control, the use of smart antennas, and security in wireless networks. He has been a PI or a project lead on projects from various DARPA programs including the Fault Tolerant Networks program, the Next Generation Internet program, and the Small Unit Operations program. He was the recipient of the US National Science Foundation CAREER Award from ANI in 2003. He also coedited the book *Ad Hoc Networks: Technologies and Protocols* published by Springer Verlag in 2005. He has served on the program committees of INFOCOM, MobiHoc, and ICC. He is the associate editor-in-chief for ACM *Mobile Computing and Communications Review (MC2R)*. He is a member of the IEEE.



Neal E. Young received the PhD degree in computer science from Princeton University in 1991. He is with the Department of Computer Science and Engineering, University of California, Riverside. His recent work has focused on fast algorithms for special classes of linear programs and on theoretical analysis of optimization problems arising in dynamic networks. He was a Hertz fellow from 1986 to 1991, an assistant professor at Dartmouth College from 1995 to 1999, and a senior research scientist at Akamai Technologies from 1999 to 2003. He is a member of the IEEE.